Proceedings of the 9th UK Conference on Boundary Integral Methods, University of Aberdeen, UK, 8-9 July 2013

REAL TIME BOUNDARY ELEMENT STRESS ANALYSIS

T.M. FOSTER, M.S. MOHAMED, J. TREVELYAN and G. COATES
School of Engineering and Computing Sciences, Durham University, Durham, DH1 3LE, UK
e-mail: timothy.foster@durham.ac.uk

Abstract. Significant improvements in the efficiency of the mechanical design process can be made through interactive stress analysis, in which re-analysis is performed in real-time on models with continuously updating geometry. Once the initial model has been constructed and solved, and matrix representations stored for re-use, the boundary element (BE) system may be rapidly modified and re-solved if the engineer chooses to apply geometric perturbations to parts of the model. The aim is to provide real-time feedback of stress contours to the engineer as the geometry is dynamically modified. Following a successful demonstration of these techniques in two-dimensions, in this work we focus on accelerating the re-analysis of small problems of three-dimensional elasticity to provide this rapid feedback for solids.

Three primary areas need to be considered to accelerate the re-solution of boundary element problems. These are re-meshing the model, updating the boundary element system of equations and re-solution of the system. Geometric changes should be accommodated whilst retaining as much of the existing mesh as possible, thus allowing the majority of the previous system of equations to be re-used for the new analysis. For the small problems under consideration, a reusable intrinsic sample point (RISP) integration scheme is shown to accelerate reconstruction of the system of equations. Significant time savings can be made in the iterative solution by preconditioning the updated system with the LU decomposition of the original system, providing a convergence rate suitable for real-time simulations. Using these techniques, real-time analysis can be achieved for some 3D simulations. It is anticipated that this can have a major impact on the way computational mechanics simulations are used in conceptual design.

1. INTRODUCTION

Real-time analysis of two-dimensional models is now a reality [1]. However, real-time stress analysis of three-dimensional models presents numerous additional challenges. A summary of previous work in this area for both the finite element method (FEM) and the boundary element method (BEM) can be found in Foster *et al.* [2]. The current work aims to extend the interactive stress analysis capability of the two-dimensional Concept Analyst [1] into the three-dimensional domain. This involves the development of innovative techniques to generate, update and analyse models. For the small problems assessed in this work, integration and construction of the BE system of equations takes a significantly higher proportion of the total re-analysis time than the subsequent solution of the resulting system. This paper summarises acceleration techniques used to reduce the construction time.

2. FORMULATION OF THE BOUNDARY ELEMENT METHOD

The boundary integral equation (BIE) can be formulated for displacements at a source point, $p \in \Gamma$, due to tractions, t_i, and displacements, u_i, on Γ:

$$c(p)u_j(p) + \int_\Gamma T_{ij}(p,q)u_i(q)d\Gamma(q) = \int_\Gamma U_{ij}(p,q)t_i(q)d\Gamma(q) \tag{1}$$

where T_{ij} and U_{ij} refer respectively to the traction and displacement kernels which form the 3×3 matrices $[T]$ and $[U]$ for each source-field point pairing.

By computing the integrations in the discretised form of (1) a system of linear equations can be derived. These are given in matrix form as:

$$[H]\{u\} = [G]\{t\} \tag{2}$$

where $[H]$ and $[G]$ contain the integrated traction and displacement kernels respectively. After applying boundary conditions, equation (2) can be rewritten in the form:

$$[A]\{x\} = \{b\} \tag{3}$$

In general the matrices $[H]$ and $[G]$ are never explicitly calculated and the integrals are instead assembled directly into $[A]$ and, with the application of boundary conditions, $\{b\}$. In the current work $[H]$ and $[G]$ will always be calculated in full as parts of the matrices are re-used during re-analysis.

3. INTEGRATION STRATEGY

To improve the efficiency of BE integration, a customised integration scheme is often computed for each element-node pairing in the model. The choice of scheme is based on the distance between the pair with specialist schemes used to deal with singular and near-singular integrals. The geometric data associated with the element: shape functions, Gauss point locations, weights and normals, are computed specifically for the custom integration scheme. For large problems it is not possible to fit all the data associated with the integration of all the elements in the model into memory; they must therefore be recomputed each time they are required.

The element is integrated according to the specified integration scheme and the associated $[H]$ and $[G]$ sub-matrices constructed. With the application of boundary conditions these are assembled to form the $[A]$ matrix and $\{b\}$ vector given in (3). Finally, the unknown singular terms on the diagonal of $[A]$ are calculated by applying rigid body motion. The complete system is then passed to the linear solver.

The number of Gauss points, m, required to integrate over each dimension of an element is a function of the size of the element, L, and the minimum distance between the element and the source node, R. In two-dimensions, L is the length of the element. In three-dimensions, L is the local dimension of the element and m must be computed independently for each dimension, producing a grid of Gauss points.

Lachat and Watson [3] developed an algorithm for finding m for quadrilateral elements. This was later generalised by Mustoe [4]. The original algorithm uses an iterative scheme to find the maximum ratio R/L; Gao and Davies [5] propose the following approximation to improve efficiency:

$$m = \frac{p' \ln\left(\frac{e}{2}\right)}{2\ln\left(\frac{L}{4R}\right)}, \qquad p' = \sqrt{\frac{2}{3}p + \frac{2}{5}} \tag{4}$$

where e is the prescribed tolerance of the relative integration error and p is the order of the singularity. The scheme is plotted in Figure 1. It should be noted that if $R/L < 0.25$ the integration order grows to infinity. If this is the case the element must be subdivided to ensure accurate integration.

An alternative scheme, based on numerical tests for surface integrals, is proposed by Bu and Davies [6]. This is generalised by Gao and Davies [5] and can be summarised as:

$$m = -0.1p' \ln\left(\frac{e}{2}\right)\left(\left(\frac{8L}{3R}\right)^{\frac{3}{4}} + 1\right) \tag{5}$$

The Bu and Davies [6] and Lachat and Watson [3] schemes are summarised in Figure 1.

In the current work a reusable intrinsic sample point (RISP) integration scheme is applied [7]. This differs from the standard scheme in that RISP uses a discrete number of integration schemes.

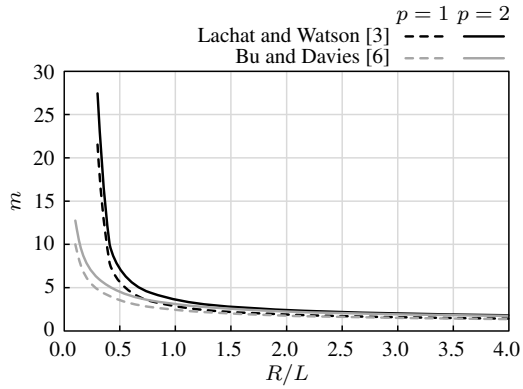


Figure 1: Integration order.

This means that the shape functions and derivatives for each integration scheme need only be calculated once for each element type. The appropriate integration scheme, and hence the number of Gauss points, m, is chosen by taking R as the distance from the centroid of the field element to the source node and L as the average side length of the element. This scheme can be applied to both quadrilateral and triangular elements.

The Bu and Davies [6] scheme runs 10% faster than the Lachat and Watson [3] scheme due to the coarser integration schemes it uses. This coarser scheme leads to a small increase of 0.1% in the L_2-norm of the stress error. However, this is permissable within the context of rapid re-analysis. Equation (5) can be rearranged to find the maximum ratio R/L for a given number of Gauss points. This is used in conjunction with the RISP integration scheme to directly determine the appropriate number of Gauss points to use across the entire element. If $e = 0.001$, this allows Table 1 to be constructed, where $\hat{m}$ is the total number of Gauss points on the element.

Singular integrals are dealt with by splitting the element into sub-elements about the singular node. Each sub-element is integrated as a quadrilateral element with two nodes lying at the singularity. However, to avoid the calculations involved in constructing the sub-elements, each element is built so that the weights and Gauss point locations associated with each sub-element are applied across the complete element so that it may be treated in exactly the same way as the non-singular elements. The total number of Gauss points used for each singular integral are given in Table 1.

Table 1: Integration schemes for $e = 0.001$.

Condition	Triangular $\hat{m}$	Quadrilateral $\hat{m}$
Corner singularity	32	64
Mid-side singularity	36	96
$0.0 \leq R/L < 0.7$	13	16
$0.7 \leq R/L < 1.1$	7	9
$1.1 \leq R/L < 2.7$	4	4
$2.7 \leq R/L$	1	1

Using a discrete number of integration schemes leads to a reduction in memory requirements and, as the models encountered in this work generally contain fewer than 2000 elements, it is possible to store all the geometric data associated with the integration in memory. The Gauss point locations, associated normals and Jacobians for all integration schemes are therefore only computed once for each element in the model. This data is stored for use in the re-analysis. The singular integration schemes are formulated such that they can be applied using the same algorithm as the standard integrals.

Changes to the architecture of the code which result in acceleration of the integration may also be made. If single rather than double precision 64bit

4. RE-INTEGRATION STRATEGY

The re-meshing algorithm used in this work is described in [2] and aims to limit propagation of geometric changes through the mesh to those areas close to the updated geometry. This means that large parts of equations (2) and (3) are unchanged. Re-integration of the system can therefore be accelerated by re-using any unchanged $[H]$ and $[G]$ sub-matrices. This is carried out by following the flowchart given in Figure 2.

An element may be updated in two different ways: it may be translated or distorted. Updates to nodes may be thought of in the same manner: Translated nodes are those that lie on translated elements; distorted nodes lie on distorted elements. Where a node lies on both types of element it is classed as translated. Where a paired element and node have the same translation applied, the associated $[H]$ and $[G]$ sub-matrices will not change. Any element or node classified as distorted will require recalculation of all the associated geometric data and sub-matrices.

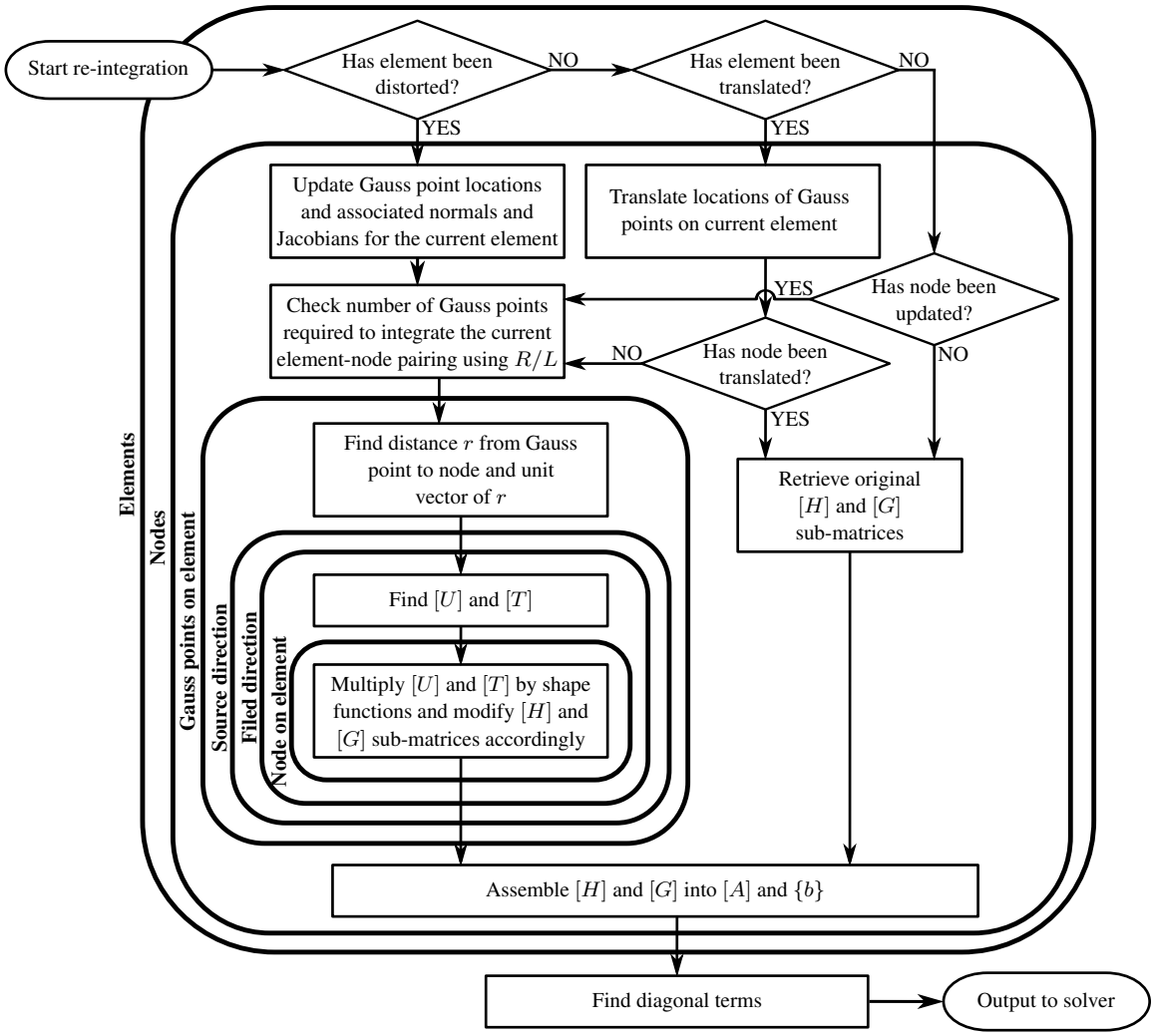


Figure 2: Flowchart of re-integration algorithm.

During re-integration, the ratio R/L is re-calculated for any node-element pairing where the node and element have moved relative to each other. If the element and node are moved further apart, this

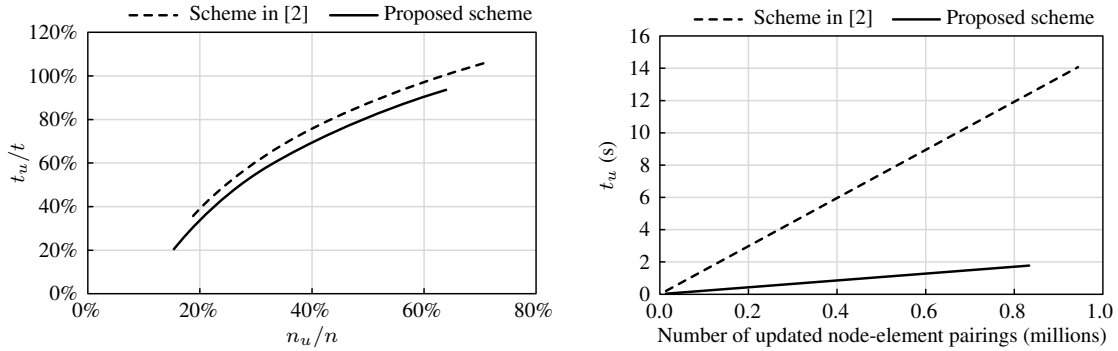
will reduce the number of Gauss points and thereby the integration time. If they are moved closer together, this will prevent the error associated with the integral from increasing. An additional refinement of the integration scheme will also be introduced whereby elements of poor quality will be re-integrated using a higher order integration scheme to reduce the potential error associated with the distortion. A detailed description of the element quality measure used in this work can be found in [2].

5. RESULTS

Figure 3(a) shows the update time, t_u, for a range of geometries with different numbers of updated nodes, n_u, for the re-integration algorithm proposed in this work and the scheme used by Foster *et al.* [2], which uses basic integration techniques without the acceleration schemes discussed in this work. However, both methods only update parts of the system that have changed. The proposed algorithm is significantly faster, providing a speed up, S, of the form:

$$S \simeq 0.004 n_u + 1.6 \tag{6}$$

The ratio t_u/t, where t is the initial integration time, is plotted in Figure 3(b) against the percentage of updated nodes, n_u/n, where n is the total number of nodes. This shows that the proposed scheme improves on the speedup achieved by Foster *et al* [2]. The affect of overheads that resulted in $t_u > t$ when $n_u/n > 0.63$ in the Foster *et al.* scheme have also been reduced. However the overheads still exist, which will result in $t_u > t$ as $n_u/n \to 1$.



(a) Absolute update time.

(b) Normalised update time.

Figure 3: Comparison of update times using the re-integration scheme.

Using single precision variables is on average 20% faster than using double precision variables. However, the size of the L_2-norm error in $\{x\}$, ε_x, associated with the reduction in precision increases exponentially with the number of nodes in the model. For the sizes of model encountered in this work single precision accuracy is sufficient. However, due to the exponential growth, double precision should be applied to larger problems if the same level of accuracy is required. It should be noted that using all single precision values in the solver routine can reduce the convergence rate, although the solution accuracy is maintained. If a few specific double precision variables are retained this degradation can be overcome.

Experimentation has shown that, when using 64-bit processing instead of 32-bit processing, a speed up of $S = 1.25$ is achieved if $n > 900$. However, for problems where $n < 450$, 32-bit processing is faster. Using the algorithm proposed in this work, models where $n > 1800$ must be computed using 64-bits as they require more than the maximum amount of memory that can be addressed using 32-bits. As $n > 450$ in the majority of problems, a 64-bit system architecture should be used wherever possible although some computers may only support 32-bits.

Figure 4 shows the effect on S as each new technique is applied to the problem. The increase in error is small when applying these schemes, $\Delta\varepsilon_x < 0.0005$.

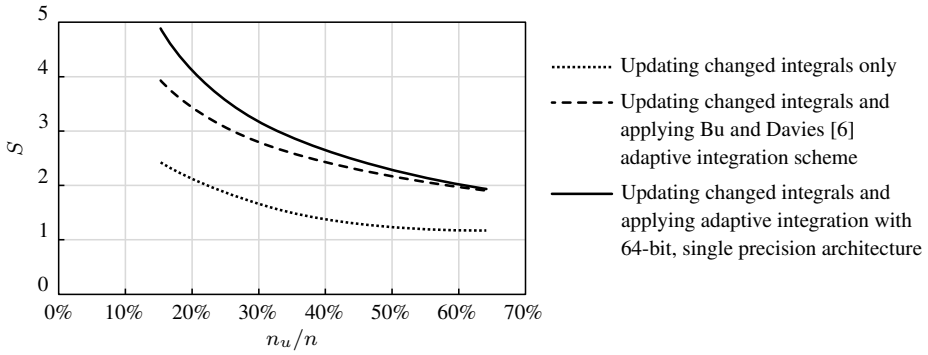


Figure 4: Acceleration of the re-integration.

6. PARALLELISATION

The first algorithm for parallelisation of the BEM appeared in 1984 in a work by Symm [8]. Davies [9] comprehensively reviews the early work on parallelising the method. Many authors, including Kane [10], use block partitioning to distribute different parts of the $[A]$ matrix over multiple processors. However the overlap between these blocks, where elements share a node, adds an extra level of complexity. Erhart [11] uses domain decomposition to solve the BE problem. This is easily parallelised as each domain can be allocated to a different processor. However, additional overheads are incurred in assessing the performance of each processor for optimal distribution of the computation.

The majority of the literature associated with parallelising the BEM deals with applying the method across multiple distributed memory devices. Here we parallelise the method on a single shared memory machine. This is a simple process, aided by the way the integration algorithm described in this work has been constructed. As the full $[H]$ and $[G]$ matrices are being stored for re-use during each subsequent re-analysis, the assembly of $[A]$ and $\{b\}$ can be carried out after the integration has been completed. This makes it possible to parallelise the integration at the top level, sending each element to a different processor. The assembly of $[A]$ and $\{b\}$ may be parallelised by generating each row of the system on a different processor.

The average speed up, S, achieved by carrying out the integration on multiple Intel Xenon X5570 2.9GHz processors on a single shared memory machine is shown in Figure 5. For the re-integration, S is 15% smaller than for the initial integration. This is due to the overheads associated with parallelisation having a greater impact on the faster re-integration algorithm. Both schemes show good scalability which is independent of the percentage of the system that is updated.

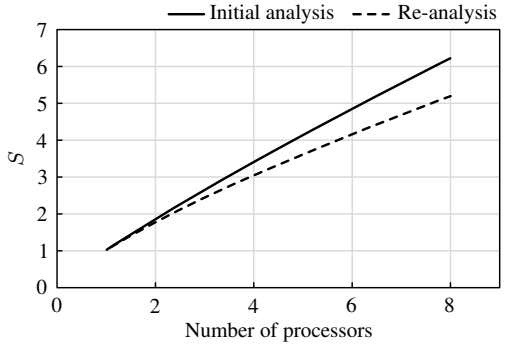


Figure 5: Integration speed gain for multiple processors.

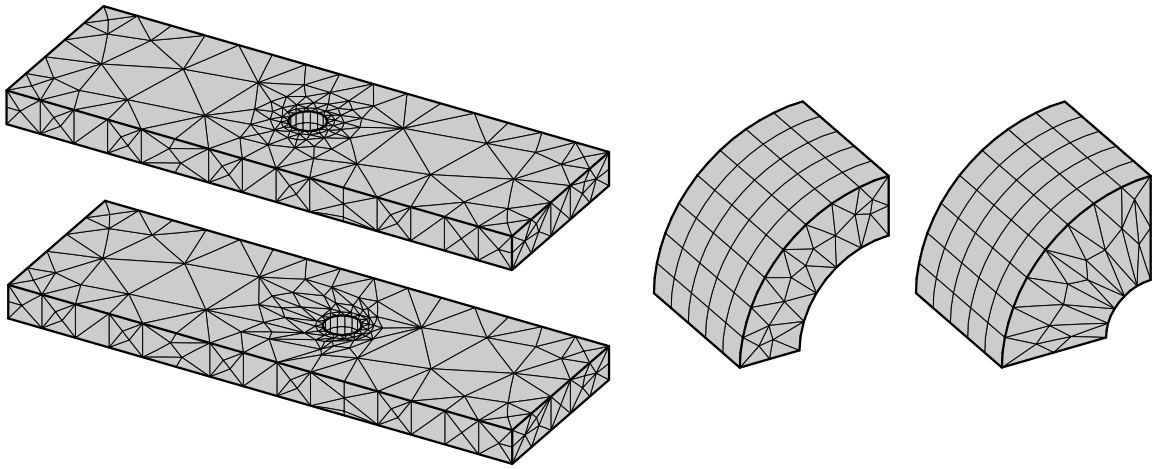
7. APPLICATIONS

The total re-solve time, T, for a problem can be given as:

$$T = t_m + t_u + t_s \tag{7}$$

where t_m is the re-meshing time, t_u is the re-integration time, using the proposed acceleration schemes, and t_s is the solve time, using a full LU preconditioned GMRES solver as described in [2].

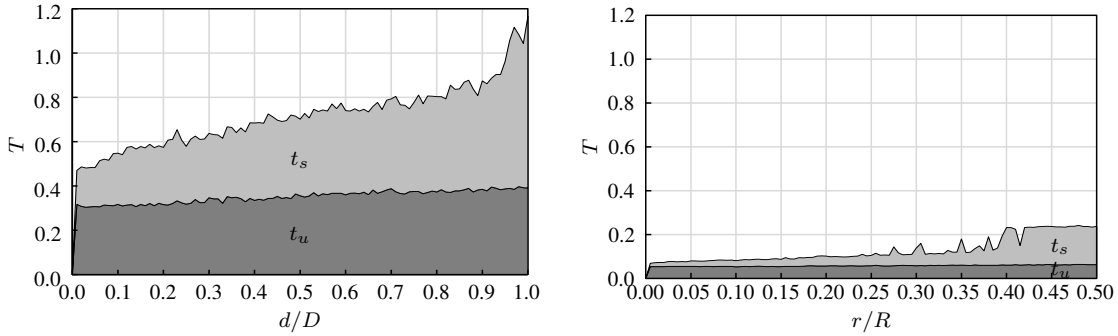
Two models are used here to assess the acceleration schemes. These are: a plate with a hole (PWH) of diameter D which is moved a distance d across the plate and a thick walled cylinder (TWC) with inner radius R which is reduced by a distance r. The PWH has been meshed with 640 quadratic elements; the TWC with 195 elements. Both models are shown in Figure 6. The benchmark integration scheme uses none of the acceleration schemes discussed in this work and is run on a single processor.



(a) Plate with a hole (PWH). (b) Thick walled cylinder (TWC).

Figure 6: Test models before and after geometric updates have been applied.

The models were assessed using all the acceleration schemes discussed in this paper and run in parallel on four processors. A summary of the re-solve times is shown in Figure 7. It should be noted that $t_m < 0.015$ seconds and has therefore not been included. For the PWH, $S = 3.6$ on a single processor, $S = 11.6$ on four processors. For the smaller TWC, S is smaller as a larger percentage of the elements are updated, $S = 2.7$ on a single processor and $S = 9.1$ on four processors.



(a) Plate with a hole (PWH). (b) Thick walled cylinder (TWC).

Figure 7: Comparison of update and solve times.

For small systems with small geometric perturbations, t_u dominates T but as the system size increases t_s contributes a larger percentage of T. The re-solution also takes longer as the model is

deformed from the original. This is caused by the associated reduction in the quality of the LU preconditioner. For the small TWC case, near-realtime analysis is achieved. The error, $\varepsilon_x < 0.03$, when compared to FE solutions for both test models. In the context of interactivity this is justifiable. However, a more refined analysis should be carried out to validate the final model to a greater accuracy.

8. CONCLUSIONS

A new, efficient algorithm for accelerating the re-integration of the BIE when a geometric change is applied to a model has been introduced and shown to be effective both in serial and in parallel. By combining the work carried out in this paper with the meshing and solution schemes used by Foster *et al.* [2] it is possible to perform near real-time re-analysis for small three-dimensional problems and significantly accelerate re-analysis of larger systems.

Acknowledgements

This research is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant EP/H000046/1. The authors would like to thank Dr. Stuart Spence of BAE Systems and Mr. Simon Walker of Jesmond Engineering for their input and support.

REFERENCES

[1] Concept Analyst Ltd., URL: www.conceptanalyst.com.

[2] T.M. Foster, M.S. Mohamed, J. Trevelyan and G. Coates (2012) Rapid re-meshing and resolution of three-dimensional boundary element problems for interactive stress analysis, *Engineering Analysis with Boundary Elements*, **36**, 1331-1343.

[3] J.C. Lachat and J.O. Watson (1976) Effective numerical treatment of boundary integral equations: A formulation for three-dimensional elastostatics, *International Journal for Numerical Methods in Engineering*, **10**, 991-1005.

[4] G.G.W. Mustoe (1984) Advanced integration schemes over boundary elements and volume cells for two- and three-dimensional non-linear analysis, *Developments in Boundary Element Methods*, (eds. P.K. Banerjee and S. Mukharjee), Elsevier, London.

[5] X.W. Gao and T.G. Davies (2000) Adaptive integration in elasto-plastic boundary element analysis, *Journal of the Chinese Institute of Engineers*, **23(3)**, 349-356.

[6] S. Bu and T.G. Davies (1995) Effective evaluation of non-singular integrals in 3D BEM, *Advances in Engineering Software*, **23**, 121-128.

[7] J.H. Kane (1994) *Boundary Element Analysis in Engineering Continuum Mechanics*, Prentice-Hall.

[8] G.T. Symm (1984) Boundary elements on a distributed array processor, *Engineering Analysis*, **1**, 162-165.

[9] A.J. Davies (1996) Parallel implementations of the boundary element method, *Computers and Mathematics with Applications*, **31(6)**, 33-40.

[10] J.H. Kane (1996) Boundary element analysis on vector and parallel computers, *Computing Systems in Engineering*, **5(3)**, 239-252.

[11] K. Erhart, E. Divo and A.J. Kassab (2006) A parallel domain decomposition boundary element method approach for the solution of large-scale transient heat conduction problems, *Engineering Analysis with Boundary Elements*, **30**, 553-563.